PUC Minas
A B A K Ó S
Instituto de Ciências Exatas e Informática

Problema dos K-centros*

Andrei Gonçalves R. Massaini e Luiz Fernando C. Rodrigues[1]

Resumo

O relatório tem como objetivo comparar duas soluções algorítmicas para o problema dos k-centros, uma abordagem por força bruta e uma abordagem aproximada, onde na primeira encontraria a melhor resposta, porém não é eficiente, pode levar tempo demais e não ser utilizada, e a segunda encontrará rapidamente uma resposta, porém aproximada e não exata. O problema dos k-centros é uma questão importante na otimização combinatória e tem diversas aplicações práticas.

O relatório compara as duas soluções em termos de tempo de execução, precisão da solução encontrada e escalabilidade para diferentes tamanhos de instâncias. Para isso, são utilizadas instâncias extraídas da biblioteca OR-library. O tempo de execução e a precisão são medidos e comparados entre as duas soluções para diferentes valores de k e tamanhos de instâncias.

No geral, o relatório fornecerá uma análise detalhada e comparativa das duas soluções algorítmicas para o problema dos k-centros, destacando suas vantagens, desvantagens e desempenho em diferentes cenários. Isso permite que os leitores compreendam melhor as características e trade-offs das soluções, auxiliando na escolha da abordagem mais adequada para diferentes contextos e requisitos.

Palavras-chave: Grafos, k-problem, k-centros Template. LaTeX. Abakos. Periódicos.

*Artigo apresentado à Revista Abakos

[1]Bacharel em Ciência da computação PUC Minas, Brasil– andrei.massaini@hotmail.com

1 INTRODUÇÃO

O problema dos k-centros é uma importante questão no campo da otimização combinatória e tem aplicações em várias áreas da computação, como logística, roteamento de veículos, telecomunicações e análise de redes sociais. Ele envolve encontrar os k pontos em um grafo que minimizem a distância máxima entre qualquer vértice do grafo e o centro mais próximo.

Neste trabalho, nosso objetivo foi abordar o problema dos k-centros no contexto da computação, desenvolvendo dois algoritmos para a solução do problema: um algoritmo aproximado e outro por força bruta.

A seguir, forneceremos uma análise detalhada das implementações dos algoritmos, juntamente com uma comparação entre eles, utilizando instâncias extraídas da biblioteca **OR-library.**

2 IMPLEMENTAÇÕES

2.1 Gerando instâncias

A função *generate_matrix_from_file*, lê um arquivo de texto que contém informações sobre um grafo, incluindo o número de vértices, o número de arestas e o valor dos centros. Cria uma matriz vazia com dimensões correspondentes ao número de vértices, preenchendo-a inicialmente com infinito. Em seguida, preenche a diagonal principal com zeros para representar que a distância entre um vértice e ele mesmo é zero.

A função lê cada linha restante do arquivo, que contém informações sobre as arestas do grafo, incluindo os vértices conectados e o custo da aresta. Essas informações são usadas para atualizar a matriz, preenchendo as células correspondentes com os custos das arestas. Como a matriz é simétrica, as células correspondentes nas posições opostas também são atualizadas.

Após a criação da matriz inicial, a função aplica o algoritmo de *Floyd-Warshall* para encontrar as menores distâncias entre todos os pares de vértices no grafo. Esse algoritmo utiliza três loops aninhados para iterar sobre todos os vértices intermediários possíveis e atualizar as distâncias na matriz. O valor da distância entre dois vértices é atualizado para o mínimo entre o valor atual e a soma das distâncias passando por um vértice intermediário.

Por fim, a função retorna a matriz de distâncias atualizada e o valor dos centros como resultados. A matriz de distâncias representa as distâncias mínimas entre todos os pares de vértices do grafo, e o valor dos centros indica os vértices considerados centros no grafo.

2.2 Algoritmos

As implementações dos algoritmos estão divididas em duas funções principais: *k_center_brute_aproximado* e *k_center_brute_force*. Ambas as funções recebem uma matriz de distâncias e um parâmetro k como entrada e retornam os seus respectivos centros encontrados, assim como o *radius* da solução.

2.3 Algoritmo de força bruta

A função *k_center_brute_force* busca exaustivamente os centros ótimos em um grafo representado pela matriz de distâncias previamente processada.

A função gera todas as combinações possíveis de K vértices no grafo. Para cada combinação, calcula-se a maior distância entre um vértice e os vértices da combinação. O conjunto de vértices que resulta na menor distância máxima é considerado como os centros ótimos.

Ao percorrer todas as combinações, a função mantém o conjunto de centros com a menor distância máxima encontrada até o momento. No final, retorna esse conjunto de centros e a distância máxima correspondente como resultado da função. Esses valores representam os centros ótimos encontrados pela busca exaustiva no grafo.

Ao realizar uma análise de complexidade concluímos que esse método é inviável para distâncias muito grandes, ou com o número k de centros superior a 2 uma vez que possui o algoritmo possui complexidade exponencial, como podemos observar na notação a seguir: O((num_vertices choose K) * K * num_vertices) Dito isso, observamos posteriormente que esse método foi incapaz de fornecer uma solução em tempo hábil para as instâncias fornecidas, sendo assim se torna necessário recorrer a abordagens mais eficientes, aproximados, que serão apresentados a seguir.

2.4 Algoritmo aproximado

A função *k_center_aproximado* implementa um algoritmo aproximado para encontrar os centros ótimos em um grafo representado por uma matriz de distâncias. O algoritmo funciona selecionando o primeiro vértice como centro inicial. Em seguida, itera sobre os vértices restantes e calcula a distância mínima em relação aos centros já selecionados. O vértice com a maior distância mínima é escolhido como um novo centro. Esse processo é repetido até que todos os centros desejados sejam selecionados. No final, o raio da solução é determinado como a maior distância entre um vértice e o centro mais distante encontrado. Esse algoritmo fornece uma solução aproximada para o problema dos centros ótimos, sendo mais eficiente do que a busca exaustiva, embora a solução obtida possa não ser a ótima. Ao analisar a complexidade do método chegamos a seguinte notação: O(k * num_vertices), com isso diferentemente da solução anterior este algoritmo fornece uma solução polinomial ao problema, embora não exata

mas eficaz para instâncias grandes.

2.5 Comparação:

A seguir, demonstraremos os resultados do raio da solução encontrado por cada algoritmo, em relação ao raio real contidos na tabela:

Instância	|V|	k	Raio	Força Bruta	Algoritmo Aproximado
1	100	5	127	TIMEOUT	190 (150%)
2	100	10	98	TIMEOUT	132 (135%)
3	100	10	93	TIMEOUT	155 (167%)
4	100	20	74	TIMEOUT	121 (164%)
5	100	33	48	TIMEOUT	72 (150%)
6	200	5	84	TIMEOUT	159 (189%)
7	200	10	64	TIMEOUT	98 (153%)
8	200	20	55	TIMEOUT	83 (151%)
9	200	40	37	TIMEOUT	59 (159%)
10	200	67	20	TIMEOUT	31 (155%)
11	300	5	59	TIMEOUT	82 (139%)
12	300	10	51	TIMEOUT	74 (145%)
13	300	30	35	TIMEOUT	60 (171%)
14	300	60	26	TIMEOUT	41 (158%)
15	300	100	18	TIMEOUT	25 (139%)
16	400	5	47	TIMEOUT	89 (189%)
17	400	10	39	TIMEOUT	57 (146%)
18	400	40	28	TIMEOUT	45 (161%)
19	400	80	18	TIMEOUT	29 (161%)
20	400	133	13	TIMEOUT	19 (146%)
21	500	5	40	TIMEOUT	53 (132%)
22	500	10	38	TIMEOUT	56 (147%)
23	500	50	22	TIMEOUT	34 (155%)
24	500	100	15	TIMEOUT	23 (153%)
25	500	167	11	TIMEOUT	15 (136%)
26	600	5	38	TIMEOUT	50 (132%)
27	600	10	32	TIMEOUT	43 (134%)
28	600	60	18	TIMEOUT	29 (161%)
29	600	120	13	TIMEOUT	20 (154%)
30	600	200	9	TIMEOUT	14 (156%)
31	700	5	30	TIMEOUT	44 (147%)
32	700	10	29	TIMEOUT	46 (159%)
33	700	70	15	TIMEOUT	26 (173%)
34	700	140	11	TIMEOUT	17 (155%)
35	800	5	30	TIMEOUT	38 (127%)
36	800	10	27	TIMEOUT	41 (152%)
37	800	80	15	TIMEOUT	25 (167%)
38	900	5	29	TIMEOUT	39 (134%)
39	900	10	23	TIMEOUT	35 (152%)
40	900	90	13	TIMEOUT	21 (162%)

3 CONCLUSÃO

Ao analisar os resultados na tabela, podemos concluir que o algoritmo de força bruta não conseguiu concluir a execução em nenhuma das instâncias fornecidas. Isso ocorreu porque o número mínimo de centros nas instâncias era igual a 5, tornando a busca exaustiva inviável em termos de tempo de execução.

Por outro lado, o algoritmo aproximado foi capaz de ser executado em todas as instâncias. No entanto, observamos uma taxa de erro considerável ao comparar o raio encontrado com o raio real das instâncias. Essa diferença percentual pode ser significativa em relação à qualidade da solução.

Com base nessas observações, podemos concluir que o algoritmo de força bruta é adequado para instâncias pequenas, onde o valor de K é inferior a 3. Nesses casos, ele provavelmente será capaz de executar sem problemas e fornecer a solução ideal. Já para instâncias com um número de centros maior que 3, o uso do algoritmo aproximado se torna essencial, pois embora apresente uma taxa de erro considerável, ele é capaz de lidar com instâncias maiores de forma mais eficiente e prática, fornecendo soluções razoáveis dentro de um tempo aceitável.